Fundamentos de IA y Machine Learning

Lección 2: Entrenamiento y Evaluación

eip ESCUELA INTERNACIONAL POSGRADOS

ÍNDICE

Entrenamiento y Evaluación 3

Presentación y objetivos 3

1. Introducción 4

1.1. Contextualización de la lección 4

1.2. Modelo y algoritmo 5

1.3. Entrenamiento y evaluación 6

2. Técnicas de evaluación 7

2.1. División en un conjunto de *train* y otro de *test, Hold-out* 7

2.2. Repetición de varias divisiones aleatorias *Train-Test* 9

2.3. Validación cruzada *k-fold* 9

2.4. Validación cruzada *Leave One Out* 10

2.5. Determinar que técnica usar 11

3. Sobreentrenamiento e infraentrenamiento 12

3.1. Sobreentrenamiento 12

3.2. Infraentrenamiento 13

3.3. Encontrar un equilibrio para evitar el sobreentrenamiento e infraentrenamiento 14

4. Métricas de evaluación 15

4.1. Métricas de regresión 15

4.2. Métricas de clasificación 17

4.2.1. La matriz de confusión 17

4.2.2. Precisión global 18

4.2.3. Sensibilidad 18

4.2.4. *False Positive Rate* 18

4.2.5. Especificidad 19
4.2.6. Precisión 19
4.2.7. *F1-Score* 19
4.2.8. Área bajo la curva ROC 19
4.2.9. Kappa 21
5. Puntos clave 22

Entrenamiento y Evaluación

PRESENTACIÓN Y OBJETIVOS

Cuando nos enfrentamos a un problema de aprendizaje automático, tratamos de generar un modelo basado en la recopilación en los datos para generar nuevo conocimiento. Para ello, las etapas de selección de la técnica, entrenamiento y evaluación de esta e interpretación de resultados se convierte en otras de las tareas esenciales del *machine learning*.

Objetivos

Al finalizar esta lección serás capaz de:

- Diferenciar entre modelo y algoritmo.
- Dividir la base de datos para un correcto entrenamiento del modelo (ajuste de sus parámetros).
- Evitar el sobreentrenamiento e infraentrenamiento.
- Interpretar las matrices de confusión.
- Evaluar un modelo a partir de distintas métricas para cada tipo de problema.

1. Introducción

1.1. Contextualización de la lección

Como se observó en la lección anterior, las fases en las que se componen la resolución de un problema de aprendizaje automático las siguientes.

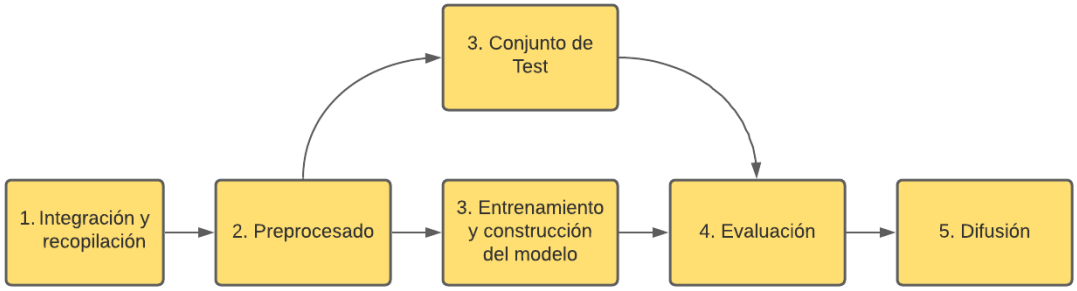

Figura 1.1: Fases de un problema de aprendizaje automático.

Tras la obtención de los datos y su importante preprocesado, es el momento de tomar una serie de decisiones:

- **Determinar el tipo de conocimiento:** predictivo o descriptivo.
- **Analizar el tipo de técnica más apropiada:** clasificación, regresión, agrupación o clustering, reglas de asociación, sistemas de recomendación, ...
- **Dentro de la técnica, que modelo es el más interesante:** por ejemplo, en clasificación tendríamos regresión logística, árboles de decisión o redes neuronales, entre otros.
- **Establecer el algoritmo de aprendizaje más adecuado:** descenso por gradiente, algoritmos analíticos, algoritmos bioinspirados, ...

Aunque ahora no tenemos todo el conocimiento para realizar estas decisiones, a lo largo de las lecciones tres, cuatro y cinco expondremos los modelos más extendidos en la literatura y los algoritmos necesarios para ajustar sus parámetros.

Esta lección se centrará en, independientemente del modelo y el algoritmo seleccionados, la fase de entrenamiento y de evaluación, interpretación y presentación de los resultados.

1.2. Modelo y algoritmo

Como se puede observar, a menudo, en la terminología del aprendizaje automático se hablará de modelo y algoritmo. Por ello, es necesario realizar un inciso y explicar estos conceptos:

> **Modelo**: un modelo es, en general, una función o estructura, que representa el conocimiento subyacente en un conjunto de datos. Dependiendo del objetivo del problema, tipo de variables de entrada, tipo de salida esperada, etc. habrá modelos más o menos apropiados. Un ejemplo sería la representación de una red neuronal, que trataremos con más detalle en la lección 4 de esta asignatura.

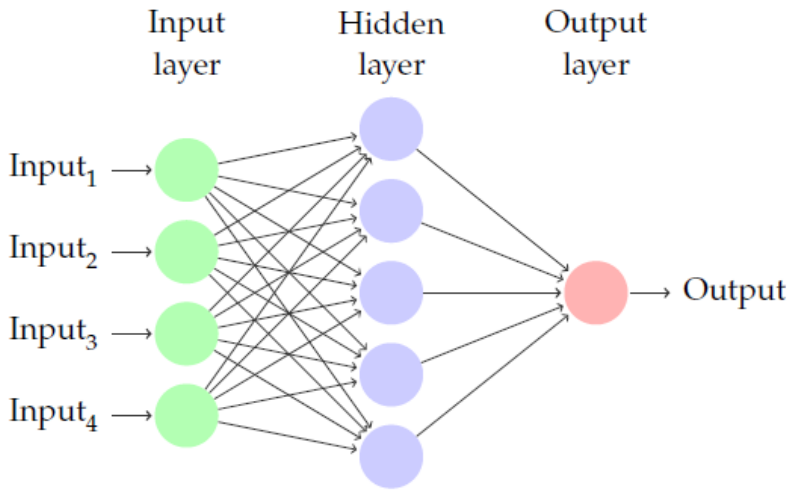

Figura 1.2: Modelo de red neuronal.

> **Algoritmo**: un algoritmo es una secuencia de pasos con un fin. De este modo, en el contexto del aprendizaje automático, un algoritmo de aprendizaje se encarga de que el modelo aprenda de los datos, o expresado de otra forma, de que el modelo se ajuste a los datos. Así, en el ejemplo anterior de la red neuronal el algoritmo trataría de encontrar los pesos de la red para hacer que la variable predicha sea lo más similar posible a la variable objetivo real.

1.3. Entrenamiento y evaluación

Para la mayoría de los modelos de aprendizaje automático existe una fase de entrenamiento. En esta etapa, es cuando aplicamos un algoritmo que ajuste una serie de parámetros del modelo, como se ha comentado previamente. El conjunto de parámetros a estimar es distinto de un modelo a otro, sin embargo, existen una serie de nociones que son comunes en el proceso.

Bajo el paradigma del aprendizaje supervisado, la idea es ajustar los parámetros del modelo de forma que la diferencia entre el valor predicho y el valor de la variable objetivo sea lo menor posible. En este sentido, podemos entrenar el modelo una y otra vez hasta que se consiga disminuir esta diferencia, o lo que es lo mismo, que el modelo se ajuste lo máximo posible al conjunto de datos.

Sin embargo, aquí surge la siguiente pregunta: ¿es justo entrenar el modelo y validarlo sobre el mismo conjunto de datos? La respuesta es no, ya que tendríamos el problema del sobreeentramiento. Aunque veremos con más detalle este concepto a lo largo de la lección, el sobreentrenamiento hace que un modelo sea muy preciso sobre el conjunto de datos con el que se ha optimizado los parámetros del modelo, sin embargo, ante un conjunto de datos nuevo, las predicciones serán terribles.

2. Técnicas de evaluación

La evaluación de nuestros modelos debe ser realizada utilizando conjuntos de datos que no han sido usados durante el entrenamiento. Para ello, las bases de datos se dividen en diferentes conjuntos dependiendo de las distintas técnicas de evaluación.

2.1. División en un conjunto de *train* y otro de *test, Hold-out*

El método más simple que se puede utilizar para entrenar y evaluar el rendimiento de un algoritmo de aprendizaje automático es dividir la base de datos original en dos subconjuntos: el conjunto de entrenamiento o *train* y el conjunto de generalización o *test*. El algoritmo realiza el entrenamiento con el conjunto de *train*, computa las predicciones en el conjunto de *test* y las compara con el valor real de la variable objetivo en este conjunto.

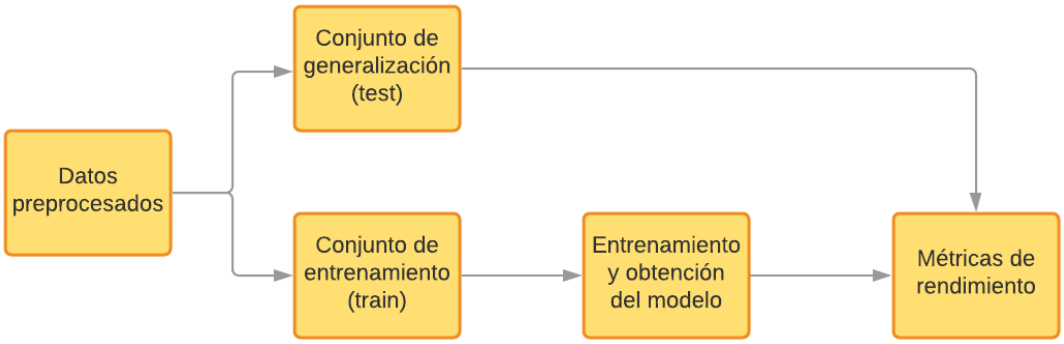

Figura 2.1: Proceso de división en un conjunto de entrenamiento y otro de generalización.

Esta técnica tiene las siguientes características:

> Se debe seleccionar el porcentaje de patrones que formarán parte del entrenamiento y parte de la generalización. Comúnmente se utiliza un 67% para entrenar y un 33% para generalizar, o bien un 75%-25%. No obstante, esto dependerá de cada problema y el modelo empleado.

- Es ideal para conjuntos de datos muy grandes donde los subconjuntos son los suficientemente representativos de la muestra original.
- El procedimiento de evaluación es muy rápido, por lo que su uso es indicado cuando el algoritmo de entrenamiento es muy lento.
- Una desventaja es que los resultados dependen de la división inicial, por lo que diferentes divisiones provocarán diferentes resultados y, en consecuencia, una gran varianza.

Al igual que ocurría en el ejemplo anterior, si se entrena el algoritmo con el conjunto de entrenamiento, obtenemos las métricas de evaluación en el conjunto de generalización y en función de estos últimos resultados decidimos si reentrenar otra vez o no el algoritmo, estamos ante la misma pregunta de que si esto es justo o no.

Es por ello, que normalmente se utiliza una pequeña división del conjunto de entrenamiento, denominada conjunto de validación. De esta forma:

1. El algoritmo es entrenado en el conjunto de *train*.
2. Se observa cual es el rendimiento en un conjunto no visto durante el entrenamiento denominado conjunto de validación.
3. En función de estos resultados, se decide volver a entrenar el modelo.
4. Se obtienen las métricas de rendimiento en el conjunto de *test*.

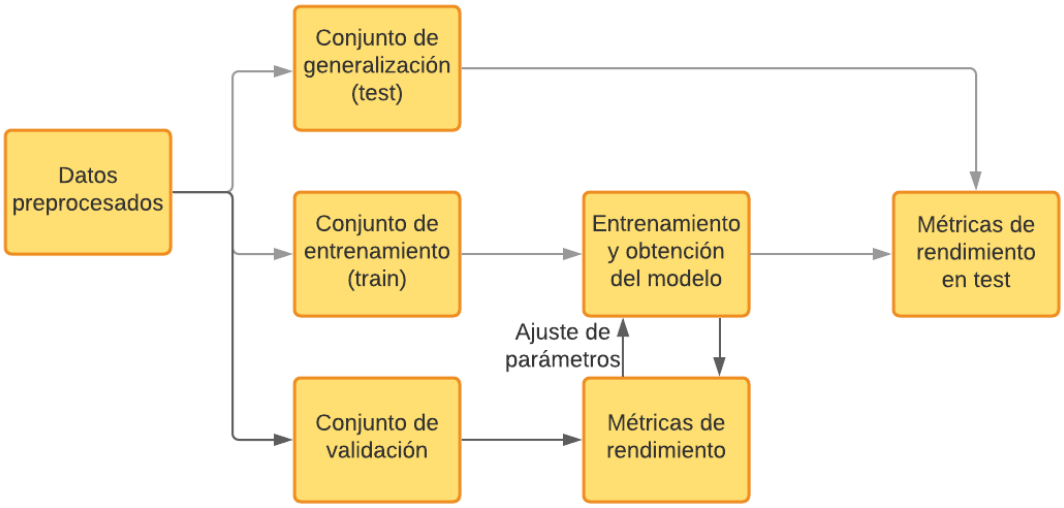

Figura 2.2: Proceso de división incluyendo un conjunto de validación.

2.2. Repetición de varias divisiones aleatorias *Train-Test*

Una solución para minimizar la varianza del modelo obtenido es repetir el proceso anterior varias veces. De forma, tendremos tantos resultados de rendimiento como divisiones *train-test* se hagan. La idea es obtener el resultado medio y calcular su desviación típica.

2.3. Validación cruzada *k-fold*

Para reducir la varianza que tiene la división en un conjunto de entrenamiento y test, también se realiza lo que se conoce como validación cruzada *k-fold*. Esta consiste en dividir el conjunto de datos en un número *k* de subconjuntos. Cada conjunto es conocido como *fold*. La idea es que se entrene el modelo con *k-1 folds* siendo el restante el destinado para generalizar. El procedimiento se repite hasta que todos los *folds* han pasado a ser el conjunto de *test*. De esta forma, la validación cruzada termina con *k* resultados correspondientes a las métricas del rendimiento de cada uno de los *folds* que ha pasado por el conjunto de *test*. Estos resultados son resumidos mediante la media y la desviación estándar de los mismos. La diferencia con la repetición aleatoria de subconjuntos en *train-test* es que cada patrón sólo pasa una vez por el conjunto de *test*.

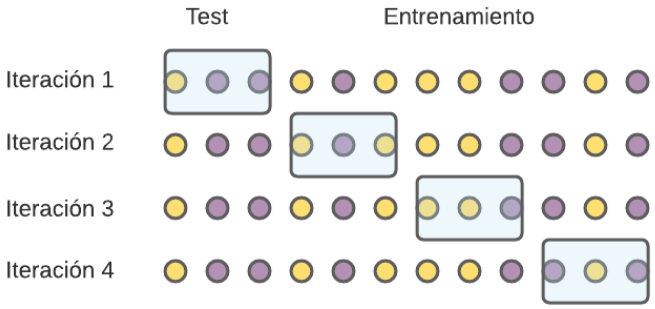

Figura 2.3: Procedimiento de validación cruzada.

Algunas de las características más importantes de esta técnica de evaluación son:

- Menor varianza que la técnica anterior.
- El algoritmo es más preciso ya que es entrenado y evaluado varias veces en subconjunto de datos distintos.
- El procedimiento es más lento que el anterior.
- Se debe elegir correctamente el número *k* de subconjuntos de forma que cada uno de ellos represente correctamente la muestra original.

2.4. Validación cruzada *Leave One Out*

Si se configura la validación cruzada de forma que el tamaño de cada subconjunto es igual a 1, o lo que es lo mismo el número de *folds* es igual al número de patrones de la base de datos, estamos ante lo que se denomina la validación cruzada *Leave One Out*.

De esta forma, se obtienen *k* resultados, referentes a la evaluación del modelo obtenido que cada iteración, con el objetivo de estimar de una forma más razonable el rendimiento de dicho modelo. Sin embargo, tiene como desventaja que se trata de un procedimiento más costoso.

2.5. Determinar que técnica usar

Las principales características de las técnicas anteriores se resumen en:

- La validación cruzada *k-fold* es considerada la mejor forma de medir el rendimiento del algoritmo en el conjunto de test. Los valores más comunes de *k* son 3, 5 o 10.
- La división aleatoria *train-test* es rápida por lo que se recomienda su uso cuando estamos optimizando parámetros con algoritmos costosos y cuando se utilizan grandes bases de datos para reducir la varianza.
- El *Leave One Out* y la repetición de varias divisiones aleatorias *train-test* pueden ser útiles cuando se pretende equilibrar la varianza en el rendimiento estimado, la velocidad de entrenamiento del modelo y el tamaño del conjunto de datos.

Aun así, todo depende del problema que estamos tratando, a partir del cuál decidiremos que metodología utilizar.

3. Sobreentrenamiento e infraentrenamiento

Ahora que conocemos el proceso completo de entrenamiento de modelos de aprendizaje automático, así como las distintas formas de dividir las bases de datos para conseguir los modelos más precisos posibles, debemos tener en cuenta dos conceptos.

3.1. Sobreentrenamiento

El problema del sobreentrenamiento consiste en que el modelo se ajusta en exceso a los datos y, por tanto, al ruido que hay en éstos. Esto resulta en modelos con un error en entrenamiento muy pequeño, mientras que el error en la generalización es bastante alto. Por ejemplo, si intentamos ajustar una curva a un conjunto de puntos bidimensionales, el siguiente ejemplo sería un buen ajuste.

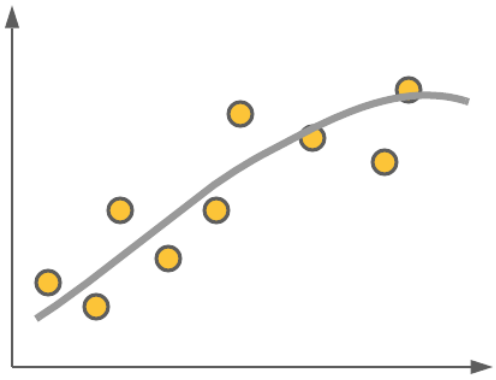

Figura 3.1: Ajuste de un problema de regresión.

Sin embargo, el efecto de un ajuste perfecto a esos puntos resultaría en un modelo con una salida de la siguiente forma:

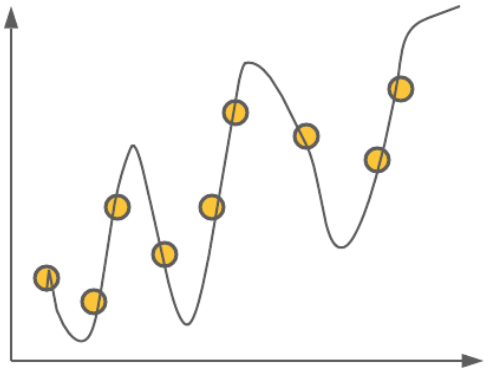

Figura 3.2: Ajuste sobreentrenado de un problema de regresión.

Ante la llegada de un dato nuevo, el error cometido sería muy grande ya que no se ajustaría bien a ese dato.

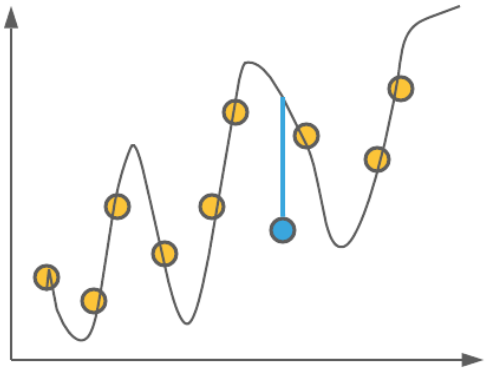

Figura 3.3: Generalización en un modelo sobreentrenado.

3.2. Infraentrenamiento

El efecto contrario al sobreentrenamiento es el infraentrenamiento, es decir, el efecto que se produce cuando el modelo no está lo suficientemente entrenado y, por tanto, el ajuste de sus parámetros sigue siendo incorrecto.

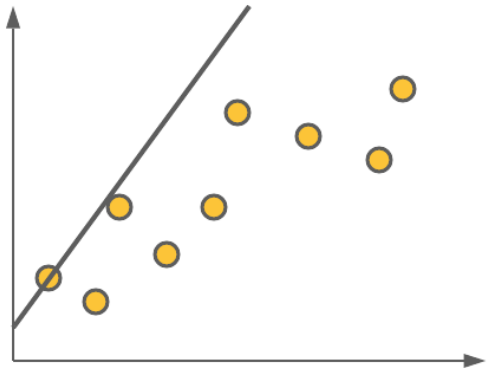

Figura 3.4: Ajuste infraentrenado de un problema de regresión.

3.3. Encontrar un equilibrio para evitar el sobreentrenamiento e infraentrenamiento

El ejemplo anterior es algo sencillo, ya que estamos representando puntos en dos dimensiones. Una mejor técnica para evitar el sobreentrenamiento es ir midiendo el error de entrenamiento y de validación a lo largo de las iteraciones en las que se va ajustando el modelo. Nunca podemos mirar el error en test, si no, como hemos indicado anteriormente, estaríamos haciendo un ajuste injusto de nuestros datos. De este modo, obtendríamos una gráfica como la siguiente, en la que hay que determinar cuando el error de validación comienza a aumentar a pesar de que el error de entrenamiento sigue disminuyendo. Es, en este momento, cuando el algoritmo empieza a sobreentrenar y es mejor finalizar el proceso.

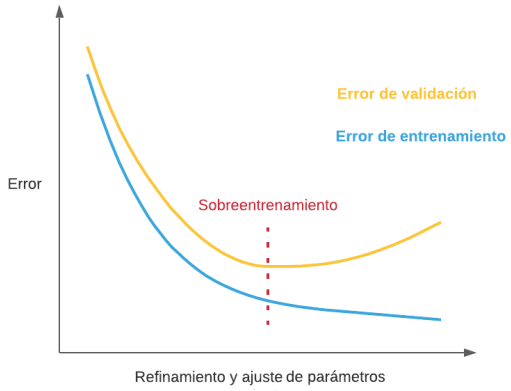

Figura 3.5: Evitar el sobreentrenamiento y el infraentrenamiento.

4. MÉTRICAS DE EVALUACIÓN

Las métricas que se eligen para evaluar los modelos de aprendizaje automático son muy importantes, ya que determinan como se mide y se compara el rendimiento de nuestro modelo con respecto a otros. Miden diversos aspectos de los modelos, por lo que un buen valor en una métrica no implica buenos valores en las demás. Se emplean para determinar la calidad del modelo o para comparar el rendimiento de varios modelos.

4.1. Métricas de regresión

Como se ha comentado previamente, si la variable objetivo es numérica y continua, estamos ante un problema de regresión. Para cada patrón $\boldsymbol{x}_i$ de un conjunto de datos $\boldsymbol{X}$, con su correspondiente valor en la variable objetivo y_i, un modelo de aprendizaje automático predecirá un valor $\hat{y}_i$. De este modo, podemos calcular las siguientes métricas error de regresión.

- **Error absoluto medio (*Mean Absolute Error, MAE*)**: es la media de las diferencias absolutas entre el valor predicho y el real. Da una idea de cómo de erróneas fueron las predicciones, y de la magnitud de error pero no de su dirección. Un inconveniente es que no penaliza los grandes errores. Así, siendo n el número de patrones del conjunto, el *MAE* se define como:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

- **Error cuadrático medio *(Mean Squared Error, MSE)***: es la media de las diferencias de los errores al cuadrado. Es muy utilizado ya que es derivable y además, penaliza los errores más grandes. El *MSE* se define de la siguiente forma:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Raíz cuadrada del error cuadrático medio ***(Root Mean Squared Error, RMSE):*** para que la medida del error esté en las mismas unidades que los valores de la variable objetivo, se aplica la raíz cuadrada al *MSE.*

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Importante

Para determinar si uno de los tres errores anteriores es un buen valor o malo, debemos contextualizar en el problema. Así, por ejemplo, un error de 10€ en la estimación del precio de una casa es un buen valor, pero es un error desastroso en la estimación del valor de un caramelo.

Coeficiente de determinación, R^2: nos proporciona una medida de calidad del modelo para predecir los resultados. Está acotado entre 0 y 1, según no se ajuste o se ajuste a los datos, respectivamente. Se calcula de la siguiente forma:

$$R^2 = \frac{\sigma^2_{y,\hat{y}}}{\sigma^2_y \sigma^2_{\hat{y}}}$$

Siendo $\sigma^2_{y,\hat{y}}$, la covarianza al cuadrado entre los valores reales y predichos, σ^2_y la varianza de los valores reales y $\sigma^2_{\hat{y}}$, la varianza de los valores predichos.

4.2. Métricas de clasificación

En los problemas de clasificación, la variable a predecir u objetivo es categórica. En resumen, las métricas de clasificación comparan si el valor predicho coincide con el valor real. Sin embargo, hay distintas métricas que nos permiten determinar el rendimiento de nuestros modelos atendiendo a distintas características.

4.2.1. La matriz de confusión

Muchas de las métricas se calculan a partir de la matriz de confusión que representa la cantidad de aciertos y fallos obtenidos por nuestro modelo para los patrones de cada clase.

Supongamos un problema de clasificación binaria, es decir, donde sólo tenemos dos clases a predecir, positiva o negativa. La matriz de confusión se resume en la siguiente representación:

		Clase predicha	
		Clase positiva	Clase negativa
Clase real	Clase positiva	TP	FN
	Clase negativa	FP	TN

- **TP (True Positive)**: verdaderos positivos. Representa el número de patrones de la clase positiva que han sido clasificados como positivos.
- **TN (True Negative)**: verdaderos negativos. Representa el número de patrones de la clase negativa clasificados como negativos.
- **FP (False Positive)**: Falsos positivos. Representa el número de patrones de la clase negativa que han sido clasificados como positivos. (Error Tipo I).
- **FP (False Negative)**: Falsos negativos. Representa el número de patrones de la clase positiva que han sido clasificados como negativos. (Error Tipo II).

4.2.2. Precisión global

La precisión global es el porcentaje de patrones correctamente clasificados. También se conoce por *Accuracy* o *CCR* por sus siglas en inglés.

$$CCR = \frac{TP + TN}{N}$$

En un problema multiclase, la precisión global se calcula como la suma de los elementos de la diagonal principal (instancias bien clasificadas) entre la suma de todos los elementos de la matriz de confusión (número total de patrones).

Es una métrica muy útil cuando la base de datos está balanceada. Sin embargo, en problemas no balanceados tiene la siguiente limitación. Supongamos un problema con dos clases, donde hay 9990 patrones de la clase 1 y sólo 10 patrones de la clase 2. Si el modelo siempre dice que los ejemplos son de la clase 1, su precisión global es del 99.9%. En este caso, la métrica es totalmente engañosa ya que nunca se detectará ningún patrón de la clase 2, que posiblemente sea la más interesante de clasificar.

4.2.3. Sensibilidad

La sensibilidad es el porcentaje de patrones positivos predichos como positivos. También se conoce como *Recall* o *TP Rate*.

$$Sensibilidad = \frac{TP}{TP + FN}$$

4.2.4. *False Positive Rate*

Porcentaje de patrones negativos predichos como positivos.

$$FP\ Rate = \frac{FP}{TN + FP}$$

4.2.5. Especificidad

La especificidad es el porcentaje de patrones negativos predichos como negativos. Es el valor complementario a la sensibilidad.

$$Especifidad = \frac{TN}{TN + FP}$$

4.2.6. Precisión

La precisión es el número de patrones positivos predichos como positivos frente al total de patrones predichos como positivos.

$$Precisión = \frac{TP}{TP + FP}$$

4.2.7. *F1-Score*

Medida que combina las métricas de Sensibilidad y Precisión. Es útil para comparar los distintos modelos. Al utilizar la media armónica, se penaliza los valores extremos, es decir, para una precisión de 1 y una sensibilidad de 0 está métrica nos diría que el clasificador es muy malo.

$$F1 - Score = \frac{2 * Precisión * Recall}{Precisión + Recall}$$

$$F1 - Score = \frac{2TP}{2TP + FP + FN}$$

4.2.8. Área bajo la curva ROC

La curva ROC caracteriza el compromiso entre aciertos y falsa alarmas. El área que queda bajo la curva, *AUC* por sus siglas en inglés, es una medida de la precisión del modelo.

- Un área igual a 1 implica que el modelo es perfecto.
- Un área igual a 0.5 indica que el modelo es tan bueno como uno aleatorio.

La representación muestra la relación entre el *TP Rate y el FP Rate*. En el siguiente ejemplo ningún modelo es consistentemente mejor que otro. M1 es mejor para *FP Rate bajos*, mientras que M2 es mejor para *FP Rate* altos.

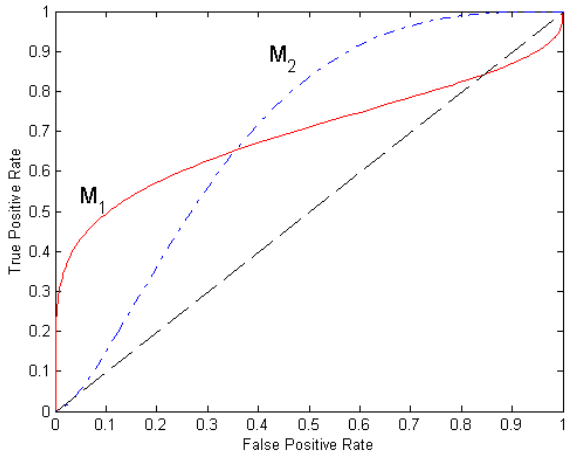

Figura 4.1: AUC de dos modelos distintos.

Importante

En problemas multiclase, estas métricas se calculan considerando una clase frente a todas las demás. La clase seleccionada será considerada como la clase Positiva y el resto como la negativa. De este modo, obtendremos un valor de cada métrica para cada una de las clases.

4.2.9. Kappa

La métrica mide el acuerdo o la relación entre los valores reales y los predichos. Se dice que es una medida más robusta que la precisión global. Puede tomar valores en el rango [-1,1], donde:

- 1 es igual a acuerdo perfecto, es decir, clasificación perfecta.
- 0 no existe relación.
- -1 es igual a un completo desacuerdo.

Se calcula de la misma manera para problemas binarios y multiclase:

$$Kappa = \frac{p_0 - p_e}{1 - p_e}$$

siendo

$$p_0 = CCR = \frac{1}{n}\sum_{j=1}^{J} n_{jj}$$

y

$$p_e = \frac{1}{n^2}\sum_{j=1}^{J} n_{j\cdot}\, n_{\cdot j}$$

donde n_{jj} es un elemento de la matriz de confusión, J es el número de clases, $n_{j\cdot}$ es la suma de todos los elementos de la fila *j*, y $n_{\cdot j}$ es la suma de todos los elementos de la columna *j*.

5. PUNTOS CLAVE

Una vez que hemos desarrollado los puntos más importantes del entrenamiento y evaluación, estaremos capacitados para:

- Diferenciar entre modelo y algoritmo.
- Particionar correctamente la base de datos para ajustar los parámetros del modelo eficientemente (entrenamiento).
- Conocer y evitar el sobreentrenamiento y el infraentrenamiento.
- Evaluar y comparar los modelos utilizando distintas métricas.
- Saber interpretar esos resultados.